HW

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/693,517	10/19/2000	Lawrence A. Crowl	SUN1P381/P4502	7922

22434 7590 06/16/2005

BEYER WEAVER & THOMAS LLP
P.O. BOX 70250
OAKLAND, CA 94612-0250

EXAMINER	
VU, TUAN A	

ART UNIT	PAPER NUMBER
2193	

DATE MAILED: 06/16/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No. 09/693,517	Applicant(s) CROWL ET AL.	
	Examiner Tuan A. Vu	Art Unit 2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *13 April 2005*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1 and 3-20* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1, 3-20* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the application filed 4/13/2005. As indicated by Applicants, claims 1,3-4,10-11are amended and claim 2 canceled.

Claims 1, and 3-20 have been submitted for examination.

Claim Rejections - 35 USC § 101

2. 35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

3. Claim 1, and 3-9 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

> The Federal Circuit has recently applied the practical application test in determining whether the claimed subject matter is statutory under 35 U.S.C. § 101. The practical application test requires that a " useful, concrete, and tangible result" be accomplished. An "abstract idea" when practically applied is eligible for a patent. As a consequence, an invention, which is eligible for patenting under 35 U.S.C. § 101, is in the "useful arts" when it is a machine, manufacture, process or composition of matter, which produces a concrete, tangible, and useful result. The test for practical application is thus to determine whether the claimed invention produces a "useful, concrete and tangible result".

Claim 1 recites a method of compilation comprising identifying instances of library object files, receiving a request to create a first instance thereof and determining whether the instance has been identified in said library object files and not creating the instance when the instance is identified in a library. The claimed step as to not creating upon some determination step leads to no concrete result being achieved. Absent a concrete, tangible, useful result, the claim fails the practical application test from above. Hence, the claim and its dependent claims 3-9 are rejected for not producing a tangible result; thus, leading toward a non-statutory subject matter.

Claim Rejections - 35 USC § 103

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

5. Claims 1, 3-7, and 9-20 are rejected under 35 U.S.C. 103(a) as being unpatentable over Burch, USPN: 6,308,320 (hereinafter Burch), in view of Fitzgerald, USPN: 5,408,665(hereinafter Fitzgerald).

As per claim 1, Burch discloses a method of compilation of source program using one or more associated depository of object files, the method comprising:

identifying one or more instances available for use in the depository of one or more object files (e.g. *reuse depository 208* – col. 10, lines 26-45; col. 11, lines 32-45) using linker symbol names (e.g. col. 10, lines 34-45) for the instance(s);

receiving a first request to create a first instance of said object file (e.g. col. 11, lines 32-45 – Note: attempt to see if new instance of object file is to be created reads on get a request to do so); and

determine whether the first instance has been identified in the one or more depository object files (e.g. col. 11, lines 32-45 – Note: checking if the object file instance is already in a depository reads on whether a first instance has been identified in such depository);

creating the first instance when the first instance has not been identified in the library object files and creating such instance when it has been identified in such library (e.g. col. 11, lines 40-47 – Note: not creating when it said instance already exists would have been implicitly disclosed).

But Burch does not specify that the depository to create a first instance of object file is library of object files. However, the purpose of library storing object files to be reused at different compilation instances was a known concept at the time the invention was made; and Burch has shown that libraries can be used dynamically with the previously stored object files at link time for creating additional or expanded object files as a result of combining reusable libraries and reusable of object files storage (e.g. col. 3, lines 55-64; libraries 114 – Fig. 1; col. 8, lines 39-42; col. 9, lines 38-47; col. 9, line 67 to col. 10, line 3). So the concept of keeping object files in a repository and the library of files reusable for linking would be equivalent; hence Burch has disclosed a form of library object files via the depository. In the hypothetical case that this concept of library object file in view of Burch's reusable store would not be clear and self-evident, this concept would have been obvious.

Analogous to Burch's approach in alleviating compiler linking resources, Fitzgerald in a method to compile and link compiled code into executable discloses the use of library of object files for scanning in order to identify which instances are needed for the final linking of the executable subsequent linking thus to obviate useless re-instantiation resources (e.g. *contained in library files, marked ... needed* – col. 9, line 32 to col. 10, line 8; Fig. 4A). Based on the concept of persisting library files needed for dynamic linkage by Burch, it would have been obvious for one of ordinary skill in the art at the time the invention was made to implement the collection of reusable object files by Burch (*reuse depository*) so it be library of object instances as taught by Fitzgerald because in conjunction with alleviating resources for having to create new object files as taught by Burch, the use of library object files of object files would enable dynamically, fast and selectively loading/activating object instances when resolving linking references (as originally intended by Burch) as evidenced by Fitzgerald, thus provide a more efficient process in which recreating of pre-stored object codes can be obviated and dynamically reused in the linking process thereby would expedite the final code linking as evidenced by the tight, fast and immediate association of library files during linking so well known in compiler technologies.

As per claim 3, Burch discloses

creating the first instance when such linker symbol name is identified as not matching any available instances in the library object files (e.g. col. 11, lines 32-45).

Although Burch does not explicitly specify that the creation of instance when the linker symbol name does not match the identified symbol names for instance available for use in the library of object files, the symbol name leading (col. 10, lines 34-45; col. 11, lines 32-45) to searching of the appropriate object file in the depository and the subsequent creation in case of no match implicitly teaches the above limitation.

As per claim 4, Burch discloses accessing a library of object files and selecting object file name referenced by linker symbol names based on what files are available in the library and saving the names (e.g. Fig. 7A, B; re cited parts of claim 1, 3 – Note: identify names that already exist implicitly disclose saving of names) but it is that Fitzgerald that discloses examining linker symbol names in symbol tables within the library files and selecting linker symbol names that are likely to correspond to instances available for use in the library of object files (e.g. *contained in library files, marked ... needed* – col. 9, line 32 to col. 10, line 8; Fig. 4A, 4C; 5A). In view of the process to identify object code instance to create anew or use without recreating as addressed in claim 1 using the symbol referencing therein by Burch and the rationale as to implement a library of object to be support linking process by Fitzgerald, the above limitation would also have been obvious for the same reasons as set forth in claim 1.

As per claim 5, Fitzgerald discloses examining and extracting linker symbol names likely to correspond to instances of library object files code (e.g. Fig. 4A, 4C; 5A); and the rationale as to using the library and the selective marking of instances by Fitzgerald combined with the selective creation of object instances by Burch would have made the current limitation obvious for the same benefits as mentioned in claim 1.

As per claim 6, Burch (in combination with Fitzgerald) disclose a linker symbol names that include predetermined sequence of characters (see Burch: *file name* - Fig. 7B, C, D; see Fitzgerald: Fig. 4A, 4C; 5A)

As per claim 7, Burch disclose hash table (Fig. 7B, C, D)

As per claim 9, Burch discloses intermediate code (Fig. 3) and Fitzgerald discloses source program in C++ (e.g. col. 4, lines 58-63). Object oriented code like C++ or Java being portable through intermediate or bytecodes was a well known concept and at the time the invention was made, implementing code with intermediate form so that it is reusable and object oriented as suggested by Burch in view of Fitzgerald C++ teaching would have been obvious for portability benefits and all pertinent advantages in OO programming language.

As per claim 10, Burch discloses a system suitable for compilation, the system comprising: a source program (e.g. Fig. 3); a depository of object files including at least one instance available for use by the program (e.g. *reuse depository 208* – col. 10, lines 26-45) an instance identifiable by a linker symbol name (col. 10, lines 34-45; Fig. 7B); an enhanced compiler suitable for compilation of source code (Fig. 3), such compiler accessing the depository object file to identify the one instance available in the library (e.g. col. 11, lines 32-45).

But Burch does not explicitly specify that the depository to create a first instance of object file is library of object files. This library limitation, however, has been addressed in claim 1 above, hence is rejected herein using the same rationale set forth therein.

As per claims 11 and 12, Burch (in combination with Burch) discloses retrieving, i.e. using an instance extractor , an instance of the library, available to be use for the program generating; and comparing one instance available with a desired instance (e.g. col. 11, lines 32-45).

As per claim 13, Fitzgerald discloses storage of an instance name (e.g. e.g. Fig. 7A, B; re cited parts of claim 1, 3 – Note: identify names that already exist implicitly disclose saving of names).

As per claim 14, Burch discloses a method of compilation using one or more associated depository of object files, the method comprising:

examining a linker symbol name of one or more associated depository of object files (e.g. col. 10, lines 34-45; Fig. 5);

extracting from the linker name one or more linker symbol names that are likely to correspond to the stored linker symbol names (e.g. Fig. 5, 7A-D – Note: a directory of file pathname being passed to a linker – see col. 12, lines 14-20 reads on a linker name table because a linker necessarily includes a internal table for effecting the resolving of symbols being passed);

storing the extracted symbol names (e.g. Fig. 5, 7A-D);

receiving a first request to create a first instance, said first instance during compilation, said instance having a first linker symbol name (col. 11, lines 32-45);

comparing the first linker symbol name with one or more stored linker symbol names; and creating the first instance when such symbol name is not yet stored (e.g. col. 11, lines 32-45 – Note: checking if the object file name instance is already in a depository reads on comparing based on whether a first instance has been identified in such depository).

But Burch d does not specify that the storage for depository of object files is library of object files of instances nor does Buch explicitly disclose examining a linker name table. This library and linker table limitations in view of the teaching by Fitzgerald using linking tables (e.g. Fig. 4A, 5A, 6A), however, has been addressed in claim 1 above, hence is rejected herein using the same rationale set forth therein.

As per claim 15, Burch (in combination with Fitzgerald) discloses a comparing without transforming the names in the library object files (e.g. Fig. 7A-D).

As per claim 16, refer to rejection of claims 7 and 9.

As per claim 17, Fitzgerald discloses a computer-readable medium to include code for implementing a method with the limitations as recited in claim 1 and claim 2, namely the steps of identifying(instances), receiving (request), determining(instance available), creating (first instance). Hence the instant claim step limitations are herein rejected (using Fitzgerald in combination with Burch) with the corresponding rejections as set forth in claims 1 and 2 respectively.

As per claims 18-20, these are computer medium claims corresponding to claims 3, 4, and 6 above, respectively, hence are rejected herein using the rejection set forth therein.

6. Claims 8 is rejected under 35 U.S.C. 103(a) as being unpatentable over Burch, USPN: 6,308,320 and Fitzgerald, USPN: 5,408,665, as applied to clam 1, and further in view of Perks et al., USPN: 6,041,180 (hereinafter Perks).

As per claim 8, Burch (combined wiht Fitzgerald's teachings) discloses obtaining a first linker symbol name for the first instance(e.g. col. 10, lines 34-45);

comparing the symbol name with those selected linker symbol names likely to correspond to object instances (e.g. Fig. 7A-C), and

creating the first instance when the first linker symbol does not match any of those selected linker symbol names likely to correspond to said instances (col. 11, lines 32-45).

But Burch does not teach template instance even though Burch code is reusable code with intermediate form (Fig. 3A) similar to object-oriented language and teaches a hash directory to be overridden (fig. 6B). Analogous to the hash comparing by Burch and the selective identification during linking by Fitzgerald, Perks via a linking process also uses comparing of object files to a encoded scheme to identify whether some discrepancies exist before assembling the final executable, and further disclose templates (col. 6, line 4 to col. 7, line 49). If Burch programming language can define templates as suggested via the overridden tables, it would have been obvious for one of ordinary skill in the art at the time the invention was made to implement Burch programming code and a override table at compile time so that templates as taught by Perks be in place to enable a visual evaluation and analysis of object modules being needed by the linking process; and thereby root out of unwanted discrepancies as implied by Perks.

Response to Arguments

7. Applicant's arguments filed 4/13/2005 regarding claims 1, and 3-20 have been considered but are not persuasive. Following are the Examiner's observations in regard thereto.

(A) Applicant has submitted that claim 1 has subject matter recited in claim 2 for complying with the 'practical application test'. But according to the analysis as set forth in the USC 101 rejection from above, this attempt to meet compliance is not persuasive because by reciting a non action no result can be inferred, and absent a tangible result, no statutory subject matter is claimed.

(B) Applicant has submitted that Burch intermediate files 122 and subsequent process (Burch col. 10, li 46-58) of identifying object files in Burch depository would not enable possible time saving as intended by the instant application (Appl. Rmrks, pg. 9 in regard to claim 1). The argument is not taking the rationale used in the rejection under perspective. The rejection has pointed out the library archive being used by the linking process in Burch would have suggested that the repository of object files can be enhanced to become dynamic library for use the linking process as evidenced by Fitzgerald, in view of the well-known concept that library can help the linking in an immediate manner for dynamic creating of final object file. The combination as set forth in the rejection would expedite the retrieval of object files and resolving of relocation of data at link time faster if those instance of object files would be used in a efficient manner as suggested by the techniques shown in Burch (not re-instantiating existing object files in persisted storage) and the techniques by Fitzgerald (not invoking object file in library that is not marked as needed). The argument by Applicant has failed to point out specific manner as to why such combined teaching, let alone the implicit teaching that Burch depository is already some form of library of files, would create adverse effects in view of known knowledge that library used a linking time help finalize the object code in immediate and non-convoluted fashion without recourse to undue resource strain. The argument about why using intermediate code and hash processing of stream would not directly support how the claim invention would distinguish from what in Burch method has been used to analogize what has been construed from one skill in the art's interpretation of the recited features from the claim. The rejection has addressed each of the limitation of claim 1; and in order for the argument to be convincing, each of these cited portions has to be shown improper, which is not the case.

(C) Applicant has submitted that Fitzgerald in the rationale used in the rejection for overcoming the deficiencies by Burch in terms of 'library object files' does not amount to any expectation of success because neither reference discuss identifying and using instances (Appl. Rmrks, pg. 10, top). The rejection has pointed out why Fitzgerald library file, or instances thereof, being selectively added based on some dependency information or marking can be used to enhance Burch's techniques for obviating useless recreation of prestored object files in light of library files usage at compiler linking time to expedite the finalizing of expanding object code. Both reference teach about how to alleviate resources by making judicious invocation of the correct instance of stored library or persisted object files to optimize the linking process. And the endeavor (i.e. how to make efficient use of stored object file/module) which both references attempt to achieve is the common ground based on which the obviousness rationale as to combine their teaching has been effected given well-known concept as to why library files support a linker process; and that is sufficient for a prima facie. Applicant has failed to point out specifically what in the cited portions of the rejection is viewed as teaching adverse or improper result as a consequence of combining Fitzgerald's library optimization technique to Burch' s efficient use of persisted object files. Applicant's arguments fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the references.

(D) Applicant has submitted that claims 10 and 14 in regard to the claimed linker symbol names have not been met by Burch (Appl. Rmrks, pg. 10, last 3 para). It is noted that Burch's method is essentially geared for incrementally and selectively compiling code based on efficient reuse of stored object files; and this reuse is done during a linking process (see col. 2, lines 35-49). There is not enough specificity in the claimed terms recited as 'linker symbol names' or 'instances' for these terms to distinguish over, respectively, the *obj* file names (Note: a name is a symbol evoked by a linking directive) being invoked at linking time (see Fig. 4, 7B, 7C) of the Burch's linking process; or the instance of file in the depository being invoked or re-instantiated for the target object file linking. Besides, a linker always have a dependency information in form of table like a relocation table or linker internal table which is considered a inherent feature (see Fitzgerald Figs. 4 for illustration; and general accepted compiler techniques mentioned by Burch -- re Aho's teaching -- col. 3, lines 38-46). The rejection has shown that by passing a name to a linker, this name has to be resolved in forms of entries in necessarily implemented linker table; and that settles the issue of linker table. As set forth in the rejection the name being passed is actually the instance fetched from a depository of object files, the so-called linker symbol names limitation reads on such object file pathname for which the linker will make an entry to resolve using a internal table as mentioned above. Hence, using inherent teaching and explicit teaching in combination, the rejection has met the limitations. The issue raised about library of object files has been addressed earlier.

For the above reasons, the claims stand rejected as set forth in the Office Action.

Conclusion

THIS ACTION IS MADE FINAL. Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (272) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (571)272-3719.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence – please consult Examiner before using) or 703-872-9306 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

VAT
June 3, 2005

[illegible]

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100